Self-Explanation through Semantic Annotation: A Survey

Johannes Fähndrich, Sebastian Ahrndt and Sahin Albayrak
DAI-Labor, Technische Universität Berlin
Faculty of Electrical Engineering and Computer Science
Berlin, Germany
Email: johannes.faehndrich@dai-labor.de

Abstract—Semantic information is considered as foundation upon which modern approaches attempt to tackle the challenges of dynamic environments – service orchestration and ontology matching are two examples for the use of such information. Yet, many developers avoid the additional effort of adding semantic information (e.g., through annotations) to their data sets – limiting the reusability and interoperability of their Apps, services, or data. This problem is called the “knowledge acquisition bottleneck”, which can be addressed by providing suitable tool support. This survey analyses the state-of-the-art of such tools that support developers in the task of semantically enriching entities. Providing an overview of available tools from the early days until now, we particularly focus on the ‘level of automation’. Concluding that automation is very limited in contemporary tools we propose a concept that mixes connectionist and symbolic representation of meaning to decrease the manual effort.

I. INTRODUCTION

ONE of the most complex construction tasks humans undertake [1, p. 1] is the development of distributed software systems that are intended to solve complex real-world tasks. Such systems, which become ever more interconnected and diverse, evolve over time. One can imagine, that this leads to heterogeneity problems, as different parties at different times make use of different technologies to reach their goals. Describing the involved entities (devices, services) of such systems in an structured and machine readable way is still an unsolved research issue and a human driven task. As the creation of these descriptions is only partially feasible at design time, many developers avoid adding semantic information to their data sets in order to save additional efforts – thus neglecting the advantages of an enrichment with additional (semantic and contextual) information. This problem is called the “knowledge acquisition bottleneck” [2].

To ease the semantic enrichment many approaches and tools are available, each one able to support developers in the task of adding semantic information to critical data. This work surveys those tools. Following the trend of leaving more and more issues dealt at runtime, we want to put a particular focus on the ‘level of automation’ that is provided by the available approaches. Hence, this work gives an overview on methods, tools and approaches, that particularly concentrates on the amount of automatism provided and the possibilities for the user to interact with the annotation process. Additionally, we focus our attention on the self-explanation property, interpreted as self-explanatory descriptions of software components as they occur in service oriented architectures or agent oriented architectures. The tools surveyed are used in creating such self-explanatory descriptions for artificial reasoners that use them during run time to couple distributed systems.

In order to identify relevant candidates, we carried out a literature research using the following databases, sources, and keywords:

- Search engines: Google Scholar1, ACM Digital Library2, IEEE Xplore Digital Library3, JSTOR4, Papers5, Springer Link6
- Keywords used: Semantic annotation tools, (semi-) automatic semantic annotation, semantic tagging, (semi-) automatic semantic tagging, ontology annotations, annotation tools for the semantic web

The conferences where crawled from the year 2000 until 2014 if available. Further interesting publications were found by specifically looking into the lists of references.

The remainder of the paper is structured as follows: Next we will introduce existing surveys and compare their results with regards to the specific focus of this work (See Section II). Afterwards, we will present the survey results, starting with the topology we used to classify all considered approaches (See Section III). Subsequently, we discuss the survey results and give some insights into future research challenges. To substantiate the results we also propose how a automatic annotation tool should be structured (See Section IV). Finally, we wrap up with a conclusion (See Section V).

1Further information: https://scholar.google.de/
2Further information: http://dl.acm.org/
3Further information: http://ieeexplore.ieee.org/Xplore/home.jsp
4Further information: http://www.jstor.org/
5Further information: http://www.papersapp.com/mac/
6Further information: http://link.springer.com/
II. RELATED WORK

The development of ontologies has a long history as they were identified timely as practical to conceptualise data [3]. Within the process of semantically annotating data the ontology defines the vocabulary and structure of the annotation result. Starting at this point, we are able to distinguish three generation of tools that support the semantic enrichment:

- The first generation are browsers, whose main purpose is viewing the semantic information, which is also called the graph of things [4].
- The second generation are annotation tools offering the capability to view and to modify the semantic information.
- The third generation are approaches that in addition to viewing and modifying the semantic information offer the option to adapt the underlying ontology.

As these generations are different in nature, the field of surveys with respect to ontology development is it also. For example, Lopez [5] and Braun et al. [6] present surveys about methodologies of ontology development. In addition, several more recent surveys describe annotation and querying tools (cf. [7], [8], [9], [10]). On the other hand, Islam et al. [11] is giving a short but holistic overview over methodologies, standards and tools for the semantic web. Covering the third generation of approaches, Ding and Schubert [12], Gomez-Perez et al. [13] and Drumond and Girardi [14] surveyed ontology learning methods. These works omit the connection to practical applications.

In conclusion, our literature research shows that although there are many surveys available, they rather focuses on methods of information retrieval than to consider the aspects of AI-methods (in particular the level of automation) in practical applications.

III. TOOLS

The aim of this section is to present an overview of the state-of-the-art regarding tools and approaches used to create and manage semantic information. To ease the reading, we will further refer to an approach, method or tool using the term solution. These solutions reach from informal “best practices” like Hash-Tags over Microformats to standards like RDFa (Resource Description Framework in attributes). The critical reader might think of this range of solutions as to broad, since the examined solutions are also classified to a property called online—meaning solutions that are able to integrate new information, enabling users to browse several information sources and collect information during runtime. To prevent the overwhelming of the user with the provided amount of information, we analyse the search capability of each solution as well. Furthermore, as some information may be of private manner the descriptive semantic information is it, too. The ability of a solution to decide which information should be shared and which should be kept secret is called privacy. One inherent feature here is the ability to share information. We refer to this feature with the term sharing. In the end, the used classification takes into account some technical aspects: Extensible and UI (User Interface). The latter one describes the way a solution represents itself to the user. Since the semantic web community provides most of the solutions surveyed in this work, the UI is mostly a web site beside some exception like frameworks. The first one—extensible—describes the ability to add new functionality to the solution. Choosing a solution that is not able to adapt to new requirement may be fatal for future work on this topic.

In order to classify the examined solutions we utilise different properties. Firstly, as the focus of this survey is based on the degree of automation, we want to classify the presented approaches based on the capability to structure unstructured information in an automated, semiautomatic and/or manual way. Hence, we distinguish the degree of automation in four categories:

1) None—None means that there is no automatism available. This implies, that all tasks have to be performed by a human.
2) Semi—Semi describes the ability to automatically perform some task with the constraint that there is still the requirement to supervise the process.
3) Collection—Collection describes the ability to automatically collect information. Since the collection of information is a time consuming task, the exploration of for example deep web annotations [17] can be automated. The extraction of structured information still requires human intervention.
4) Full—Full means the capability to collect information, extract additional information (e.g. annotations) as well as integrating new information into the information source without any human intervention. This is not considered restricted by the possibility of a manual annotation, but rather to propagate and integrate the newly gained information.

Besides the automation aspect, we want to clarify whether a solution is platform independent or not. This is important to the heterogeneous character of hard- and software used in smart environments (and the semantic web). Here, we also have to take into account the used language, which describes the semantic information. As mentioned above, there is the trend to leave more details dealt at runtime. Hence, the examined solutions are also classified to a property called online—meaning solutions that are able to integrate new information, enabling users to browse several information sources and collect information during runtime. To prevent the overwhelming of the user with the provided amount of information, we analyse the search capability of each solution as well. Furthermore, as some information may be of private manner the descriptive semantic information is it, too. The ability of a solution to decide which information should be shared and which should be kept secret is called privacy. One inherent feature here is the ability to share information. We refer to this feature with the term sharing. In the end, the used classification takes into account some technical aspects: Extensible and UI (User Interface). The latter one describes the way a solution represents itself to the user. Since the semantic web community provides most of the solutions surveyed in this work, the UI is mostly a web site beside some exception like frameworks. The first one—extensible—describes the ability to add new functionality to the solution. Choosing a solution that is not able to adapt to new requirement may be fatal for future work on this topic.

The classification of the examined solutions is illustrated in Table I. In order to give some more structure to the results, we proceed with a description of the considered solutions based on their essential functionality and classified according to the three generations mentioned above.
We refer to this generation of solutions as browsers. The main purpose of browsers is viewing the graph of things, which is manifested in annotations that are attached to the web of documents. Berners-Lee et al. [50] introduce the Tabulator browser.

<table>
<thead>
<tr>
<th>Solution</th>
<th>Language</th>
<th>Platform ind.</th>
<th>Online</th>
<th>Search</th>
<th>Privacy</th>
<th>Sharing</th>
<th>Extens.</th>
<th>Automation</th>
<th>UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Disco [18]</td>
<td>RDF</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Collection</td>
<td>Web</td>
</tr>
<tr>
<td>2 MSpace [19]</td>
<td>RDF</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>3 RDFa Developer [20]</td>
<td>RDFa, Micro</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>4 Flamenco [21]</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>5 Onk [22]</td>
<td>RDFs</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>Collection</td>
<td>Web</td>
</tr>
<tr>
<td>6 Longwell [23]</td>
<td>RDF</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>7 Sigma [24]</td>
<td>RDF</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>Collection</td>
<td>Web</td>
</tr>
<tr>
<td>8 Aquabrowser [25]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>None</td>
<td>Web</td>
<td></td>
</tr>
<tr>
<td>9 Freebase Parallax [26]</td>
<td>RDF</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>Collection</td>
<td>Web</td>
</tr>
<tr>
<td>10 Tabulator Extension [4]</td>
<td>RDFs, OWL</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>None</td>
<td>–</td>
</tr>
<tr>
<td>11 RangeAnnotator [27]</td>
<td>RDF</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>12 GrOWL [28]</td>
<td>OWL</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>None</td>
<td>Forms</td>
</tr>
<tr>
<td>13 OntoStudio [29]</td>
<td>multiple</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>Semi</td>
<td>Forms</td>
</tr>
<tr>
<td>14 Swoop [30]</td>
<td>OWL</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>Semi</td>
<td>Web</td>
</tr>
<tr>
<td>15 GKB Editor [31]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>None</td>
<td>Forms</td>
</tr>
<tr>
<td>16 Melita [32]</td>
<td>XML</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Semi</td>
<td>Forms</td>
</tr>
<tr>
<td>17 OBO Edit [33]</td>
<td>OBO</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>18 Magpie [34]</td>
<td>RDF</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>Collection</td>
<td>Web</td>
</tr>
<tr>
<td>19 DOME [35]</td>
<td>RDFs, OWL</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>Collection</td>
<td>Web</td>
</tr>
<tr>
<td>20 Biggy Bank [16]</td>
<td>RDF</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>21 Semantic Turkey [27]</td>
<td>RDF</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>Semi</td>
<td>Web</td>
</tr>
<tr>
<td>22 UIMA Web Anno. [36]</td>
<td>RDF, OWL</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>Semi</td>
<td>Web</td>
</tr>
<tr>
<td>23 Haystack [37]</td>
<td>RDFs</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>24 IBM EODM [38]</td>
<td>RDF, OWL</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>25 Topia [39]</td>
<td>RDF</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>Collection</td>
</tr>
<tr>
<td>26 Protege [15]</td>
<td>RDF, OWL</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>Collection</td>
<td>Web</td>
</tr>
<tr>
<td>27 Scooner [40]</td>
<td>RDF</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>28 Morla [41]</td>
<td>RDF</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>None</td>
<td>Forms</td>
</tr>
<tr>
<td>29 CmapTools O.E. [42]</td>
<td>RDF, OWL</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>30 Chimarr [33]</td>
<td>RDF, DAML</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>31 KAON2 [43]</td>
<td>OWL, DL</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>Semi</td>
<td>–</td>
</tr>
<tr>
<td>32 Knowd [44]</td>
<td>RDF, OWL</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>33 Virtual Ontology Modeler [45]</td>
<td>RDFs, OWL</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>34 Ontolingua [46]</td>
<td>KIF</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>None</td>
<td>Web</td>
</tr>
<tr>
<td>35 Mobi [47]</td>
<td>OWL</td>
<td>×</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>Collection</td>
<td>Web</td>
</tr>
<tr>
<td>36 OntoGen [48]</td>
<td>RDFs, OWL</td>
<td>×</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>×</td>
<td>×</td>
<td>Semi</td>
<td>Forms</td>
</tr>
<tr>
<td>37 MnM [49]</td>
<td>KMi</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>–</td>
<td>×</td>
<td>Semi</td>
<td>Web</td>
</tr>
</tbody>
</table>

A. The 1st Generation

We refer to this generation of solutions as browsers. The main purpose of browsers is viewing the graph of things, which is manifested in annotations that are attached to the web of documents. Berners-Lee et al. [50] introduce the Tabulator browser. Here, the semantic meta information about some resource is collected and displayed in tables. Tabulator allows the user to search through the presented information and to group them by sources; but not to modify them. Several other solutions for viewing semantic information are available. They can be subsumed under the term semantic browser (e.g., [4], [19], [21], [23], [25], [51], [52], [53], OpenLink Data Explorer\(^7\), Zigist\(^8\), Marbles\(^9\)). In this work we put particular interest on methods on editing the semantic information. Therefore, we can neglect most of the first generation solutions. However, on behalf of the interested reader, we can refer to surveys which focus such solutions (cf. [7], [8], [9], [11]).

As a first step towards a broader function range, Disco [18] additionally used the index Sindice\(^10\) to collect semantic information online. Following a similar approach, Sigma [24] automates the collection and consolidation from multiple information sources and is focused on collecting and viewing the entities resulting from a query. Although, the main purpose of first generation solutions is viewing the information, this does not mean that there is no automatism. The Aquabrowser [25], for example, indexes the information made available to it and creates bags of words and facets without human intervention. Here, the interested reader is pointed to Stepf aner et al. [54], which introduce a taxonomy of faceted search. Following a different approach, the Freebase Parallax [26] solution can be classified as set-based browser. This type of solution allows switching between properties collected in sets [54]. These approaches are illustrated in Table I at the positions 1–9.

B. The 2nd Generation

The restriction of read only solutions, leads us to the second generation—namely annotation solutions. Second generation solutions have the capability to modify the semantic information. Tools like GrOWL [28] and Knowd[44] offer the

\(^7\)For further information, refer to http://ode.openlinksw.com

\(^8\)For further information, refer to http://dataviewer.zitgist.com/

\(^9\)For further information, refer to http://marbles.sourceforge.net/

\(^10\)For further information, refer to http://sindice.com/
capabilities to view and edit semantic information described within ontologies. They can be classified as prototypes of the second generation. Due to the wealth of such solutions, we will further describe only those introducing new functionalities.

Berners-Lee et al. started tip-toeing towards the editing of public semantic information with Tabulator Redux [4]. In their work, the discuss where the semantic information should be stored. Ignoring privacy issues, Tabulator Redux enables user to add semantic information to a public wiki. Following Ciravegna et al. [32], it seems reasonable that users should create annotations, as they can annotate their points of interest at nearly no cost (on-the-fly). Furthermore, the authors introduced different requirements that must be accomplished and that are tackled by their own solution. Melita [32] addresses multiple usability issues arising from the pro-activeness of the user and separates the annotation process into two phases: The training phase, where the user adds annotation manually and the active annotation phase, where the system adds semantic information automatically. During the training phase the user is supported by the learning algorithm (LP)$^2$ [55], which enables an automated annotation behaviour. A similar solution is represented by Amilcare [56] also using the (LP)$^2$. An additional feature is introduced by Magpie [34]. Magpie allows the use of ontologies to annotate elements of websites. Furthermore, it enables user to specify services associated with the annotation entities. This functionality leads to the automation of ontology development by leaving the architecture open for new services. Chimera [33] describes another aspect of the creation and maintaining of semantic information. The authors argue that ontologies should be created in a distributed manner and propose approaches to maintaining and merging semantic information in ontologies. The DERI Ontology Management Environment [35] (DOME) is a specialized ontology editing and maintaining concept, focused on a ‘community-driven ontology management’. Hence, the focus lies on alignment, versioning and aggregation. As DOME focuses on the distributed maintaining of semantic information an automatism has been established to populate information. These approaches are illustrated in Table I at the positions 10–20.

C. The 3rd Generation

The second generation solutions allow the user to import existing ontologies for further use. Missing here is the capability to extend these ontologies, which leads us to the third generation of solutions, which are context-aware. Meaning that these solutions are able to adapt the used ontology to the context of use. Here, Pacienza et al. [27] introduce the Semantic Turkey, an extension of the Firefox browser, which was originally developed as a semantic bookmarking tool in 2007 [57]. In a further development stage, it was combined with the RangeAnnotator [27] enabling the extraction of information encoded in RDFa and Microformats. The extracted information are integrated into an UIMA$^{11}$ process. In addition, the RangeAnnotator adds the capability of Xpointers$^{12}$. Another solution based on the Semantic Turkey framework, presented by the same research group is STIA [58], an annotation tool to organise pertinence between laws. Fiorelle et al. [36] present an additional extension of the Semantic Turkey named UIMAST Web Annotator. Here, structured information as in HTML or PDF documents can be annotated and used to enrich user defined ontologies. This process is called Computer Aided Ontology Development (COD). Consequently, their approach is proposed as COD Architecture (CODA) with the goal of semi automatic ontology creation. It is open to extensions during runtime by using the OSGi$^{13}$ standard. Following a similar approach, Scooner [40] integrates several information extraction techniques to boot strap concepts out of a knowledge base. OntoGen [48] extends this automatism using multiple artificial learning approaches that support the user during the creation process by proposing comparable concepts of existing ontologies.

After having created tools to work with ontologies, the semantic web community fostered their technologies to feed back into semantic tools like Haystack [37]. Haystack uses RDFa to describe functionalities and user interfaces with the goal to create web applications. The crux of Haystack lies in the orchestration of services producing the functionality in the background and presenting their results to a user. The CmapTools Ontology Editor [42] describes the formalisation problem of unstructured information to structured information in a concept map-based manner. Furthermore, they distinguish between expert, experienced and normal users by adapting the user interface to ease the introduction phase to the user.

In contrast to previous solutions, frameworks exist which offer extensive features for the development of ontologies as e.g., the EMF Ontology Definition Metamodel [38] (EODM). One can imagine, that there are solutions that can not clearly be marked as frameworks for developers or as development suits for the creation of ontologies without any programming. Protege [15] and its counterparts Ontosaurus [59] and the Generic Knowledge Base Editor [60] can be located between both worlds. Another research challenge is addressed by the Topia [39] project. Here the use of semantics is discussed within the generation of hypermedia [61]: ‘The Topia project is developing a system that generates presentation structure around media objects returned from semantic-based queries.’ [39]. Therefore Topia offers capabilities to combine informations from multiple sources concerning one topic using ontology matching techniques. With the Modeling Wiki (Moki) [47] a solution for user generated content is presented, which allows to extend a semantic wiki with formal ontologies. These structured descriptions can be interpreted as self-explanatory, depending on the amount of information modelled as formal semantics.

One of the most advanced annotation frameworks is created with MnM [49]. After a manual annotation, the MnM

$^{11}$Unstructured Information Management Architecture (UIMA) - http://uima.apache.org/

$^{12}$For further information, refer to http://www.w3.org/TR/xptr-framework/

$^{13}$Open Services Gateway Initiative Framework (OSGi) - http://www.osgi.org/
framework is able to annotate new documents automatically. Although, MnM is based on a rather specific ontology language (KMi) it stores its annotations in an ontology and is able to extract annotations automatically after a learning phase.

It seems that with the advancement in this research, the goal of creating self-explaining elements is getting into reach. These approaches are illustrated in Table I at positions 21–37.

IV. Discussion

In our survey we analysed approaches that allow for the observation and editing of semantic information. Based on this survey we can state that many tools have emerged in the semantic web community. As it was our intention to classify analysed tools by their ‘ease of use’, we want to put up the respectively identified ‘level of automation’ for discussion. To start with, none of the examined approaches was able to work in a fully automated fashion. Thus, we can emphasise that there is still a difference between the stated aims of semantic research and the reality. In our opinion, semi-automatisms or solutions that are capable of learning can be considered the bleeding edge. However, any semi-automation involves human interaction, which implies that user interfaces have to be provided [62]. In our opinion, sharing semantic information is another very promising concept, however, this mechanism puts another issue in focus: privacy. Admittedly, privacy is an important issue whenever data is made available, yet, matters of privacy are far beyond this work. We leave such considerations open for future works and endorse the concept of sharing semantic information as a very capable one. Using technology independent standards for the description of semantic information may additionally further the acceptance for this mechanism.

Whenever information sources are updated (either by means of annotations or by automated procedures), the speed at which the updated information become available, plays an important role. If the update occurs (almost) immediately, we refer to the process as being ‘online’ capable. Online capability allows users to make annotations while browsing data sets. This feature may foster semantic annotation processes to be a natural part of browsing. Furthermore, when it comes to the Internet, finding and retrieving data can be considered as a constituting functionality. However, an ever increasing amount of information makes this task difficult and fosters the use of semantic enrichment of datasets. We therefore argue, that tools have to account for sophisticated search routines. Referring to our main intention, that is, to identify promising tools for further extension, we want to conclude at this point.

Taking the above mentioned properties into account, the general trend in this research area becomes fairly apparent. To foster the (automated) derivation of self-explaining information, approaches such as UIMA Web Annotator (CODA) seem to be worth extending. Admittedly, the current version of CODA is still miles away from the stated aims of semantic research, where ‘everybody might say anything about anything’ [4], yet, in our opinion, CODA is the most promising approach to achieve this goal.

A. Research Challenges

An AI that should be able extract sense or meaning from texts requires the ability to learn new meaning by itself and, thus, requires the ability to explain new words to itself. We defined this ability in a prior work [63] and within this work substantiated that there are still many hurdles that must be addressed to achieve this objective:

Meaning itself need to be represented in an appropriate way (in a formal manner) to be handled by an AI. Since meaning is not precisely defined, this is subject to research. We will look at meaning in the linguistic sense, which can be defined as follows: Meaning is what the source of an expression (message) wanted the observer to infer from the expression [64]. Since semantics is the theory on how meaning is transferred, a semantic transference and interpretations process is required. There are four parts for the meaning of a word which are of concern to an AI:

- **Denotation**: The so called denotation represents the primary or basic meaning of a word. This can be seen as the definition of a word that is represented in some kind of mental lexicon (or a dictionary).
- **Connotation**: The connotation is the abstract idea presented by the word. This can be seen as the conceptual representation of the meaning of a word. This includes the connectionist interpretation of meaning since here the meaning is interpreted as the unity of its relations to other concepts.
- **Conceptualisation**: To be able to come up with a conceptual representation of the meaning of a word, one needs to abstract from the word to a specific concept (i.e., one needs to connect the word with a known concept). This process is named Conceptualisation and helps to clarify a word within a language.
- **Pragmatics**: The meaning of words is not independent of the context the words are used in. Thus an context dependent representation of meaning (a pragmatic one) has to be created (e.g. mouse (computer) vs. mouse (pet)).

Furthermore starting from the meaning of one word, the meaning of sentences need to be extracted. We neglect this here, since it is seen as a next step after having a meaningful representation of a single word.

Technically adding semantic information generally rises the question on how to make this data available: publicly available or with restricted access. Firstly, semantic information might be directly attached to the respective dataset. However, this implies the source to be editable, which furthers the idea of some ‘semantic information service’ and transfers the accessibility issue to the owner of such service. On the other hand, additional semantic information may be stored locally and thus foster distributed (information) networks. The question on how to manage such data (especially in terms of accessibility) remains a topic of research. Secondly, in order to store semantic information, an adequate syntax has to be selected. It is difficult to mention a universal solution for this purpose as any potential scheme has to be expressive.
One challenge here is to select the right definitions of the word\textsuperscript{14} from the utilised datasources to be used in the decomposition.

This semantic network is used to spread activation or pass markers through the network.\textsuperscript{15} This is denoted by the different colours and markers in Fig. 1. The marker (represented as chips next to each node in the depiction) might carry symbolic information that steers the activation spreading. To be able to react to different markers, each node in the semantic network has a node interpretation function reflecting its behaviour. The node interpretation function inflects how the node processes incoming markers, how he passes outgoing marker on to other nodes and if he is activated. In this way, e.g., a “NOT” node passes its markers to the next node so that this one activates its opposites (in linguistic named antonyms). Since semantic relations like synonym and antonym relations have different meanings as well the relation interpretation function allows to specify how a relation passes on markers. In this way symbolic information like temporal logic can be encoded in the network. One challenge at this step is the amalgamation of the connectionist representation in the semantic network and the symbolic representation provided by a node and edge interpretation function.

During the activation through priming we can influence how the amalgamation of symbolic and connectionist representation of meaning is contextualised. By activating the right concepts out of the context, the marker passing will activate different nodes in the semantic network and thus contextualise the representation of meaning. Here the selection of parameters and concepts to activate is challenging. Finally we need an interpretation of the output of the marker passing to extract the meaning represented.

The automatic annotation then can be done by activating the word we want to annotate in the semantic network using the generated ontology of the marker passing for the annotation. If we want to annotate the word ‘Bank’ in a text discussing the financial crisis, the activation will have a stronger activation on ‘Bank’ as an financial institute then on the seating accommodation. This is because the priming will probably use words like money, accounting, currency or equivalents from the text during the activation. Thus the approach is able to annotate the text with context dependent meaning.

Regarding the proposed concept on how an automatic annotation component could be build, we want to extend the definition of Fähndrich et al. [66] of self-explaining system as follows:

\textit{Definition 1:} A self-explaining system is able to create an internal knowledge representation of an unknown concept in a pragmatic manner through the use of external information sources and communicate the so-created meaning to other systems.

\textsuperscript{14}This challenge is related to the word sense disambiguations and is one reason for the need of contextual information during the decomposition.

\textsuperscript{15}Marker passing subsumes activation spreading since the classical activation spreading can be modelled with a marker that carries the activation level as numeric value.
Definition 1 has two parts: The first part requires the system to be able to explain new concepts to itself which means to create a denotation and a connotation in a manner that the system can reason upon this internal knowledge representation. The second part describes the ability to communicate this meaning to a other system in a manner that the other system is able to create its internal representation.

V. Conclusion

This work provides an overview on approaches, methods, and tools that support developers in comfortably viewing, editing and/or adding semantic information to relevant data. In doing so, we put particular emphasis on the inherent requirements of self-explaining systems. One important requirement here is the level of automation. Besides this and in order to classify the examined solutions several other properties were introduced. However, we focused our survey on approaches that automatically collect and add semantic information mainly at the applications runtime and distinguished the level of automation into four different increasing categories. To sum up, we can say that their are only a few solutions available that offer (semi)automation capabilities. These solutions use, for example, learning algorithms to support users during the annotation process. Nevertheless, most of the examined solutions did not focus automation and we are far from fully automated annotations. In order to clarify the research progress here, we discussed the results of the survey. Substantiated by this discussion we revealed the limitations and formulated research challenges/questions that must be answered by the community. Here, beside the main question of how automatism can be realised, it might be interesting to discuss how much automatism is wanted respectively needed to create self-explaining systems and system components. The results of the survey neglected the authors thought of an existing fully automated approach. With the goal to improve the state-of-the-art, we presented unsolved research challenges and plan to exercise some of them. Here, we will select and extend a fitting solution and try to increase the degree of automatism. However, at the very first, we want to discuss and formulate a reasonable and formalised definition for self-explaining systems.

VI. Acknowledgement

This work is being supported by the German government in the Bundesministerium für Wirtschaft und Energie (BMWi) Project: "Erweiterte und adaptive Elektromobilitätsdienste: Technologie, Entwicklung, Bereitstellung", FKZ: 16SBB007A.

References
